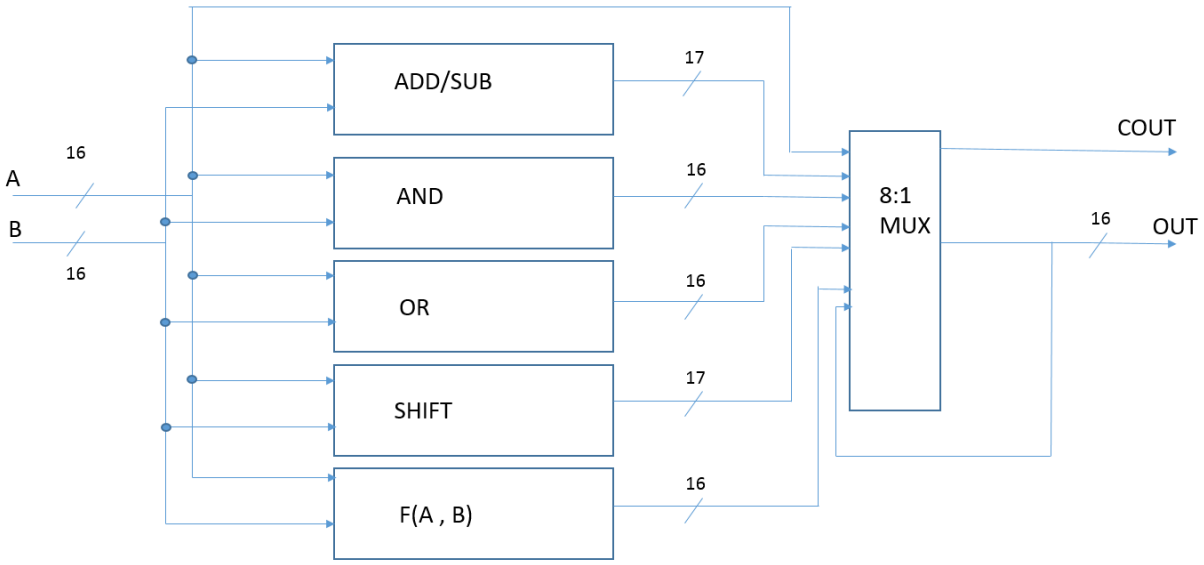
ECE3663 Design Project:
Design Review #1

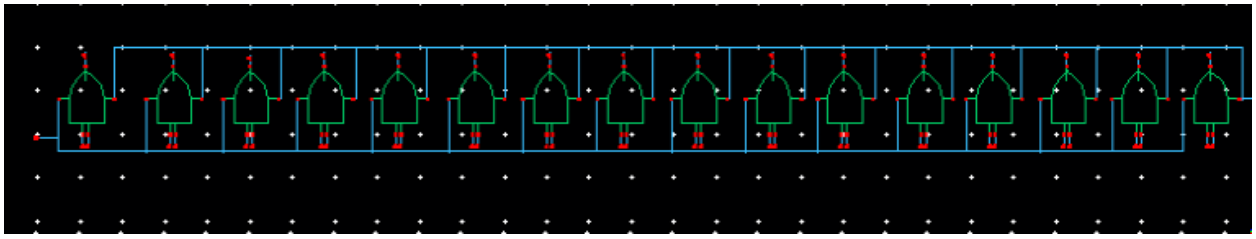
General Overview:

For the first stage of the project, we designed four different components of the arithmetic logic unit. First, schematics for each component were created at the transistor level. Then, these subcircuits were placed in parallel to create larger circuits that could perform bitwise operations on 16-bit inputs. When discussing how the 'PASS A' function should work, we came up with two possible ways to implement it into the ALU. The first would be to simply have all of the 'A' inputs connected to a dedicated input of the multiplexer. The second would be to add a set of pass gates, possibly connected between 'A' and the output, that could be turned on and off to implement the 'PASS A' function. To implement the multiplexer, we started by designing a 2:1 MUX using four NAND gates. Six of these 2:1 MUXes were connected together to create the 1-bit, 8:1 MUX (connections shown in schematic below). We followed the general approach shown below, using several components in parallel to complete the chosen task. We combined the add and subtract methods in an effort to reduce overall space and complexity. Both the 'ADD/SUB' block and 'SHIFT' blocks produce an extra bit that is used to determine the 'COUT.' We have several ideas for our arbitrary function, including a multiplier function and a XOR function. The multiplier block, while powerful, would consume quite a bit of power and could possibly be too complex for this assignment. The XOR function would be simpler and easier to implement, but we would need to decide if it is worthwhile to include. Before the next design review, we need to settle on a single function to implement and begin the process of designing the function to determine how it will be integrated into our ALU. We also need to decide on an implementation of the 'PASS A' function, as we do not have a concrete implementation of it at this time. Furthermore, we need to start thinking about how to design the registers we will need for the second review, and what type of add/subtract architecture we will use. Once these tasks are completed and the basic structure of our ALU is determined, we will begin optimizing our design to improve desirable metrics. We have a solid foundation but need to determine new and creative ways to better the metrics of our design. We want to reduce power usage, decrease delay, and also decrease area used. We will use the following formula to determine our success: Metric=(Active Power)*Delay^2*Area. With the components we currently have, we will update our methods to reduce these parameters and increase our system's efficiency.

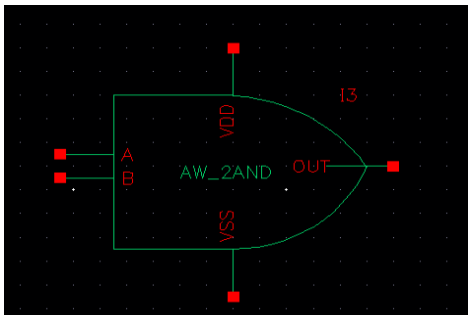


AND Block Implementation:

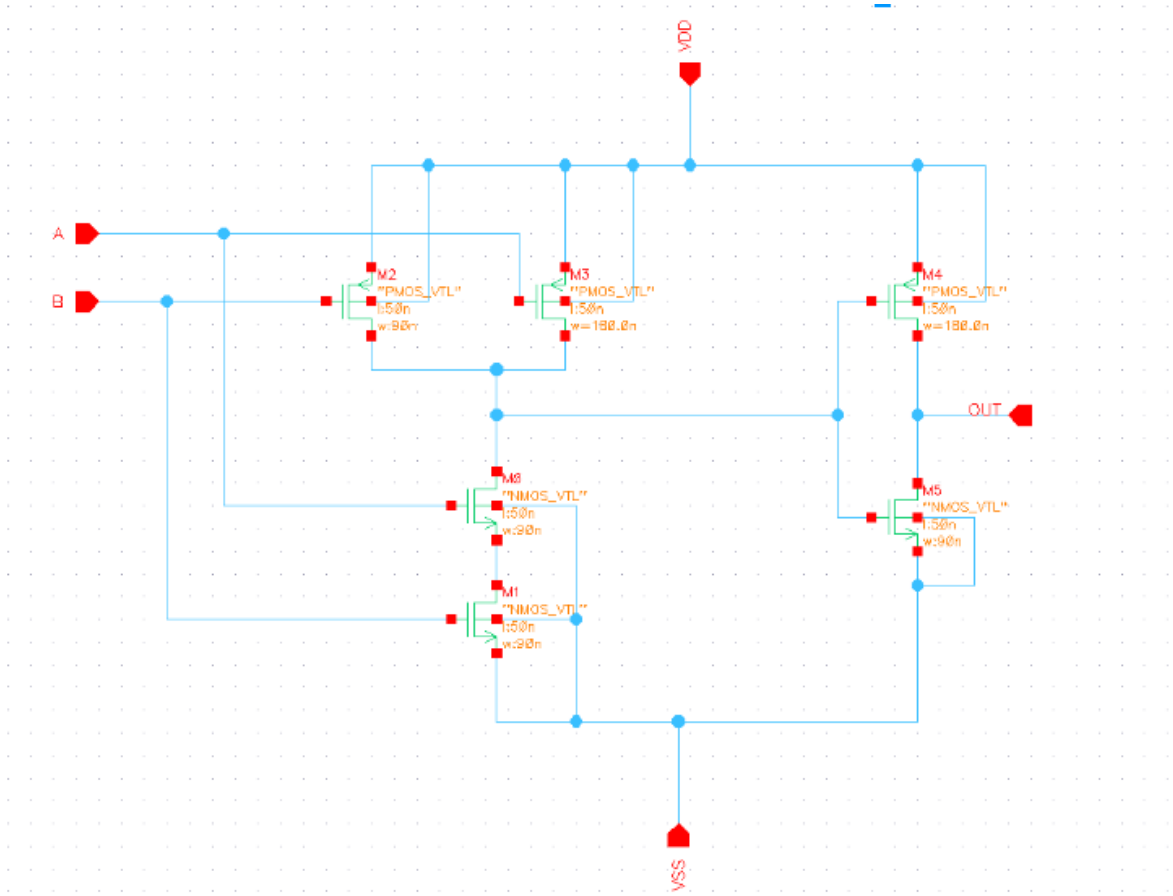
Using the basic sub-circuits designed in the previous assignment, we used a series of parallel AND gates to form a 16-bit wide AND gate. Shown below, the gate acts as a bit-wise AND device.



Each of the gates above consists of the symbol shown below.



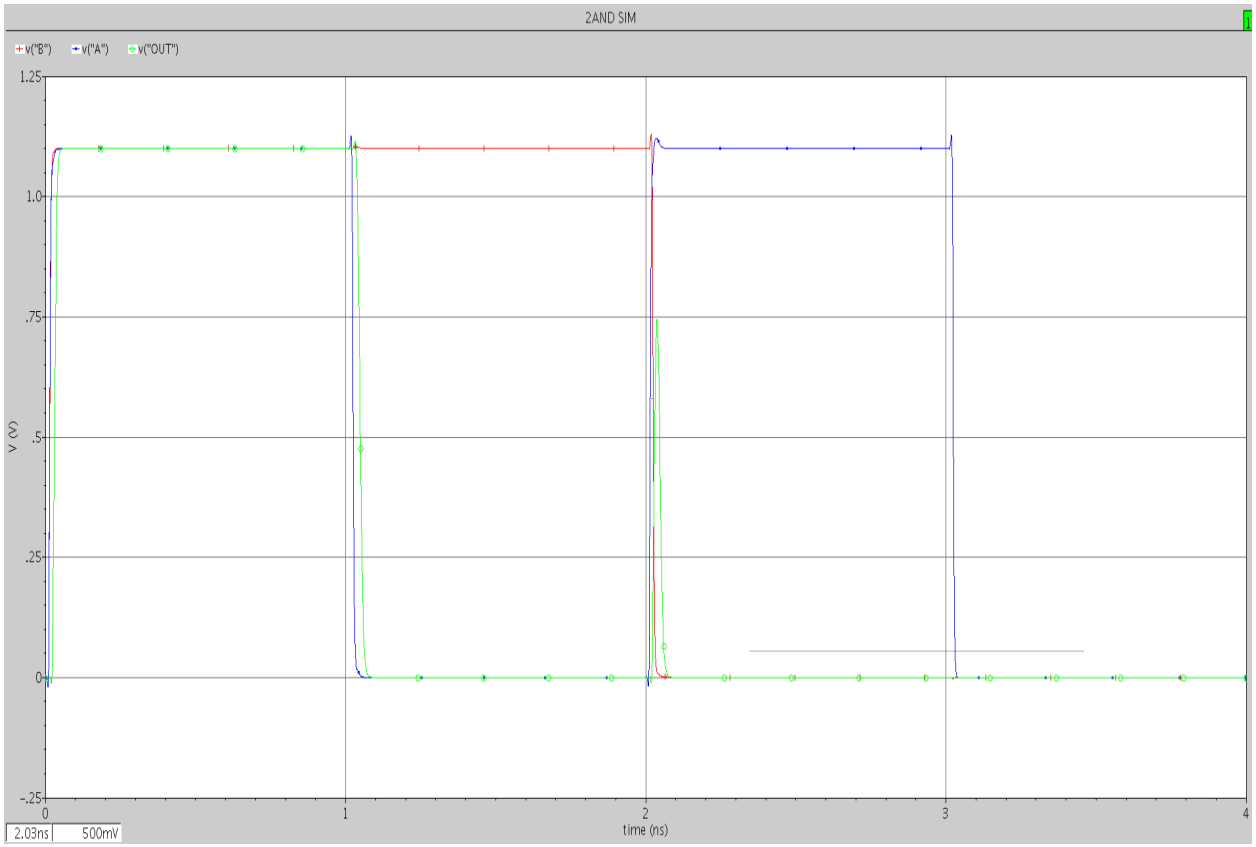
Each of these gates is, in turn, composed of the following gate-level schematic:



2-Input AND Simulations:

To ensure the AND gate performed the proper function, we tested each of the possible input combinations.

We used one simulation to show the various output conditions. This was done by having periodic inputs where the period of input B was twice that of A, this allowed for the A input to toggle twice during each B position (i.e. VDD and ground), covering all possible input conditions.

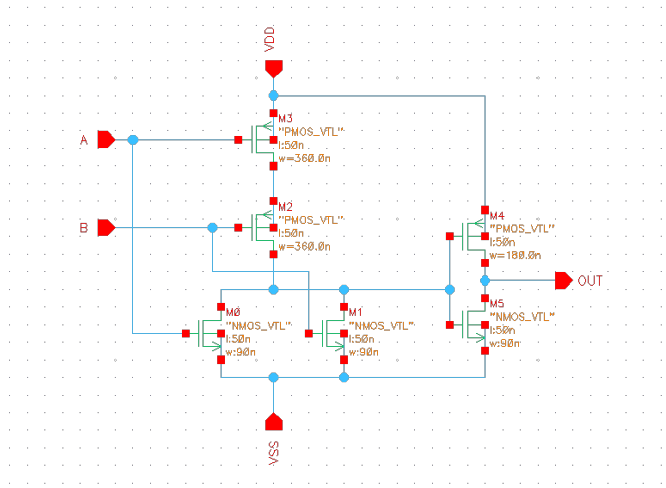


[Green = OUT , Blue =A , Red=B]

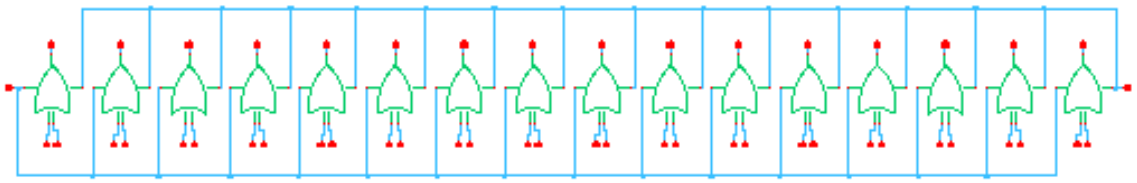
OR Block Implementation:

Using the basic sub-circuits designed in the previous assignment, we used a series of parallel OR gates to form a 16-bit wide OR gate.

Transistor level schematic of our two input OR gate:

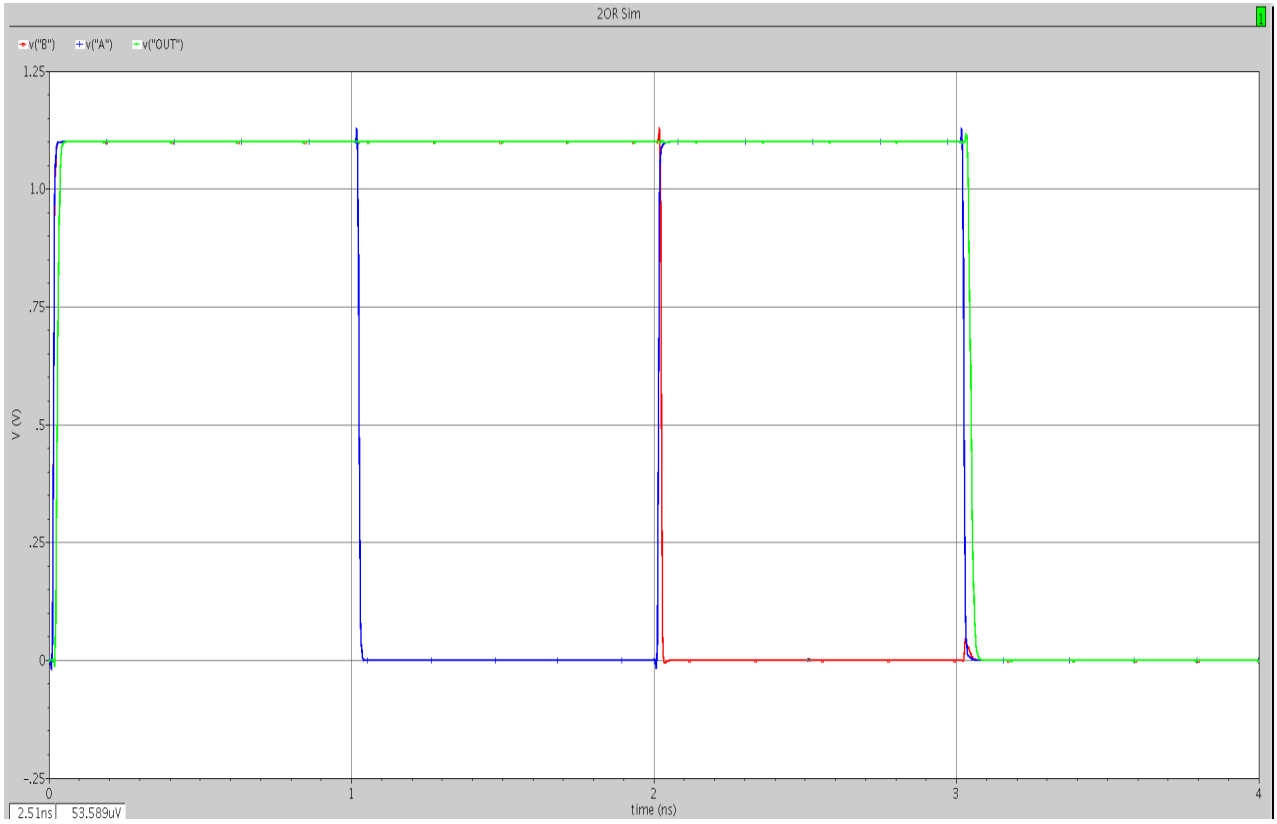


Sixteen OR gates connected in parallel to form a 16-bit, bitwise OR gate:



2-Input OR Simulation:

Following the same approach as the 2-Input AND, we chose to use two periodic inputs with offset periods. The following graph shows the results:

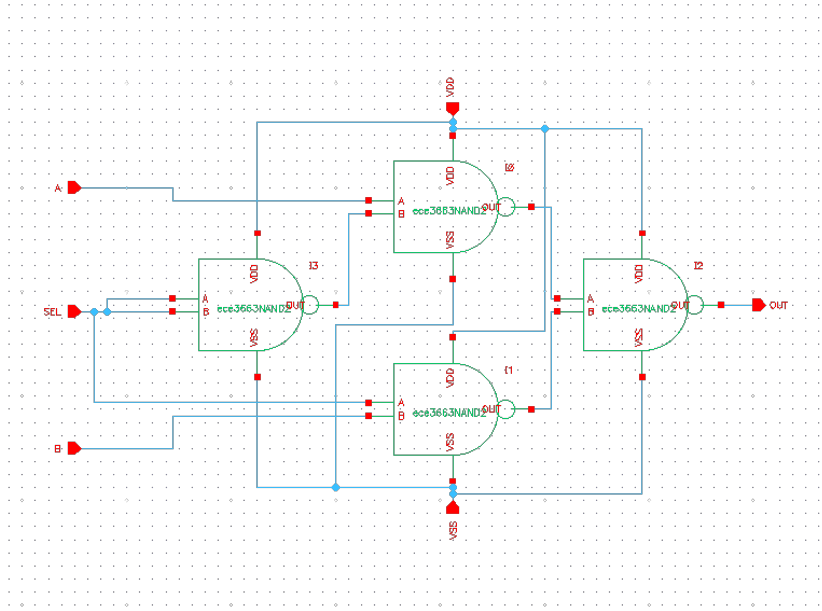


[Green = OUT , Blue =A , Red=B]

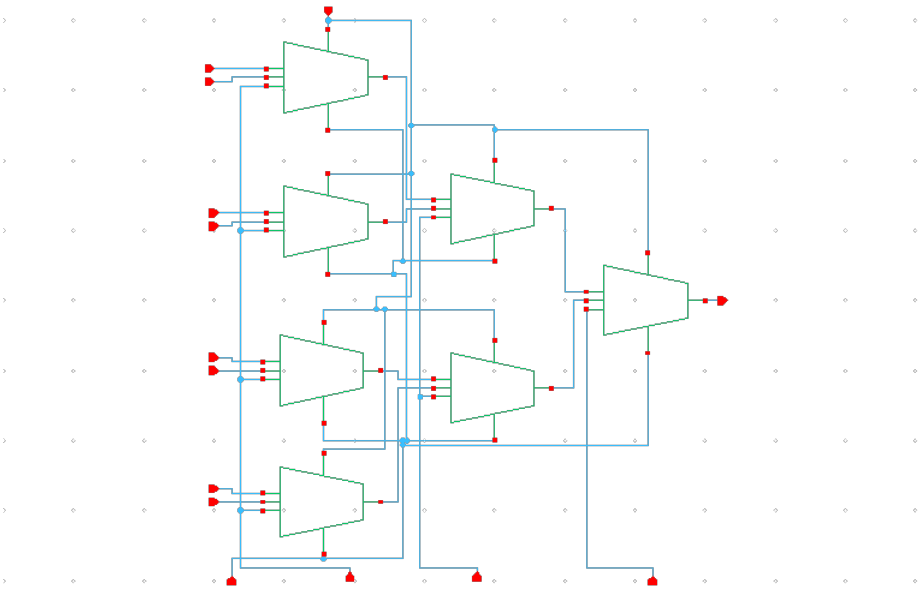
MUX Implementation

Building the 8:1 MUX involved first creating a 2:1 MUX, and then chaining six together to make the 8:1 MUX.

Below is the gate level schematic for our 2:1 MUX. The first NAND is used as inverter.

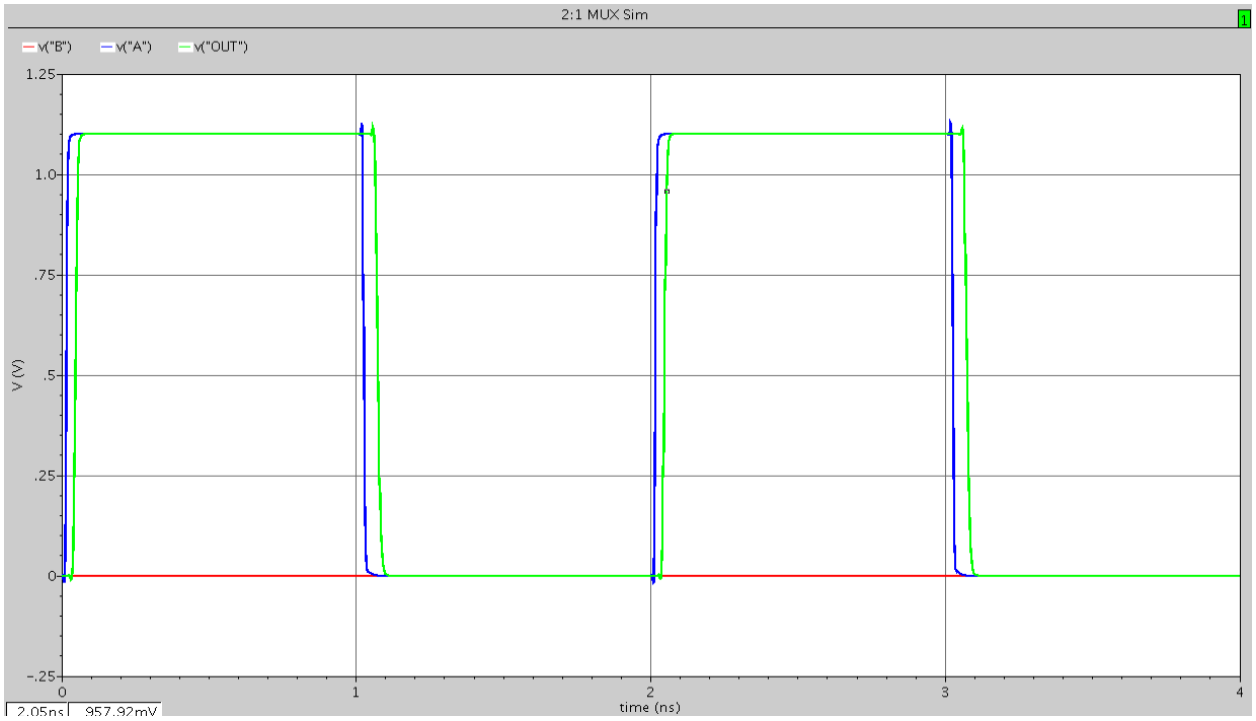


Six 2:1 MUXes connected to create an 8:1 MUX. The three rightmost pins at the bottom are the select lines.

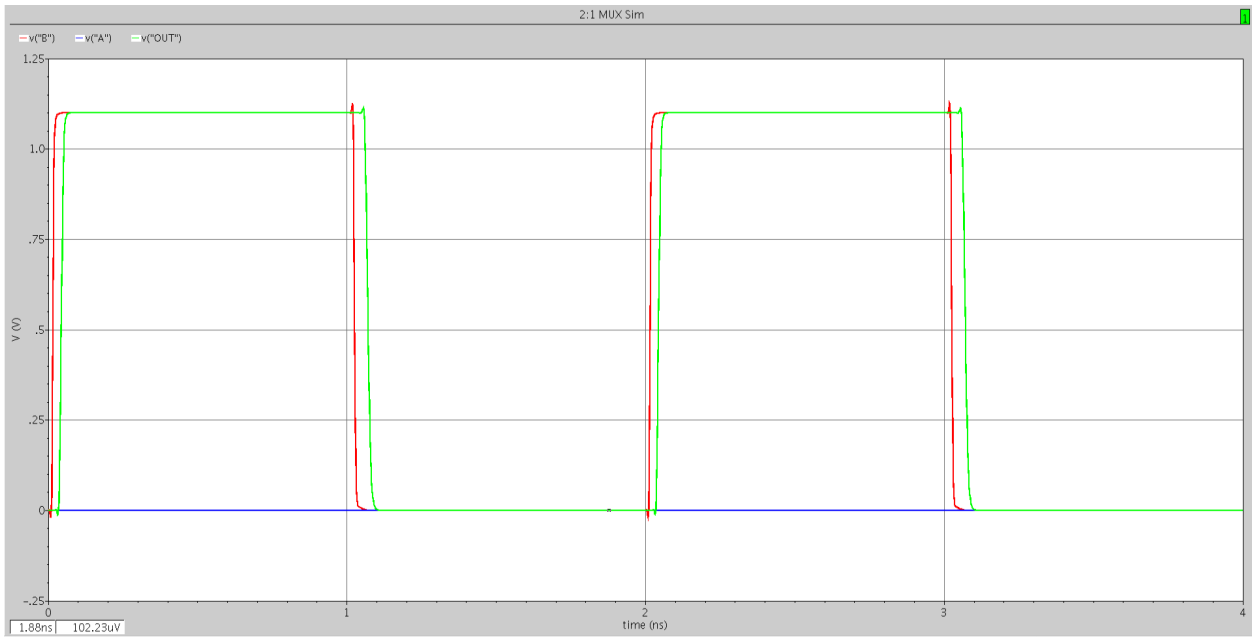


2:1 Mux Simulation:

We used a series or two simulations to show the mux's operational characteristics. In the first simulation, the control line and input B are held low while the selected input, A, is toggled to show that the output follows input A. In the next simulation, input A is held low, the select line is held at VDD, and input B is toggled to show that the output follows input B.



[Green = OUT , Blue =A , Red=B, Control =Ground]



[Green = OUT , Blue =A , Red=B, Control=VDD]

Pass Gate

A possible implementation of our pass gate, if we decide to use it.